発表タイトル：

視空間に基づいたビジュアルサーボによる
Eye-And-Hand 型ロボットの線形近似式を用いた制御

発表者名：広瀬僚平

発表日：2018 年 10 月 25 日

Q. 近似値と真値の差はどこまで許容するのか.

A. 求めた近似式を実際のサーボに用いた際に，用いる前と同じ結果になるかで判断します．本研究は，point-to-point の『リーチング制御』であるため，最終的にアームの手先が目標の手先・位置に到達しておれば許容するものとします.

Q. 近似式を線形式とした理由.

A. 先行研究において視空間と関節空間に線形性があると示されていたので，視空間座標から関節角を求める式も近似式として線形式を採用しました.

Q. SD の意味.

A. SD は式(１)によって求めることができます．よって SD は，『近似によって得られた関節角』と『真値である関節角』とのばらつきを表す数値となります．したがって，SD が小さいほど近似精度が優れていることが分かります.

$$SD = \sqrt{\frac{1}{n-1}\sum_{q_1,\cdots,q_6}(\boldsymbol{q}'-\boldsymbol{q})^T(\boldsymbol{q}'-\boldsymbol{q})} \quad \cdots(1)$$

ただし，

$$\boldsymbol{q}\ \left(\text{真値である関節角}\right)=\begin{bmatrix}q_1\\ \vdots\\ q_6\end{bmatrix}\quad \boldsymbol{q}'\left(\text{近似した関節角}\right)=\begin{bmatrix}q_1'\\ \vdots\\ q_6'\end{bmatrix}\quad n:\text{要素数}$$

である.

Q. グラフの重なりは視覚的判断のみか.

A. 手先の軌跡の類似性は視覚的判断のみです．グラフの重なりで確認した方が直感的に精度の善し悪しが判断できると考えたためです．視覚的判断以外には数値

評価として SD(ばらつきを表す数値)を参考にします.

Q. 線形近似できない条件の理由と対抗策.

A. 線形近似できない理由としては，出力が関節角 6 つの 6 自由度であるのに対し，入力が視空間座標(γ， θ， δ)の 3 つの 3 自由度であるから，近似精度が落ちているためだと考えられます．例えば，一部の関節角の変域を狭めることで，精度が良くなるという現象が起こることがあります．これは，その関節角をほぼ動いていないとみなせ，自由度が擬似的に減少したためであると考えています．

また対抗策として，入力の自由度を増やすということが考えられます．実際の制御では入力が 9 自由度なので，入力を 3 から 9 自由度にした場合の比較を図 1 に示します．また，ばらつきを表す SD の値を表 1 に示します．

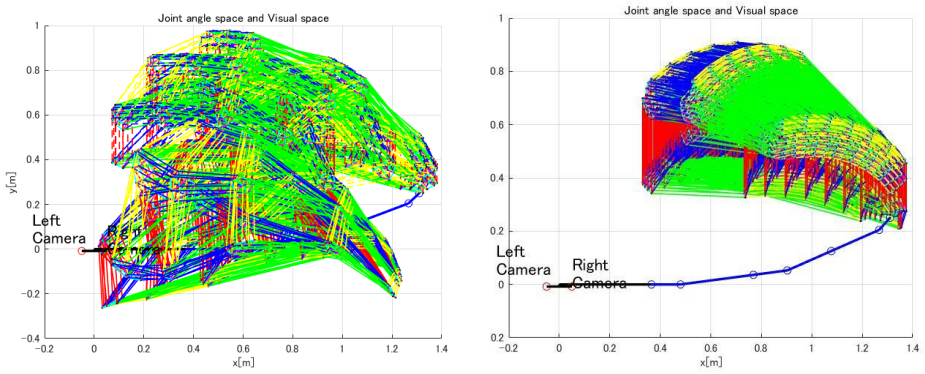

図 1　入力の自由度を変えた際の変化(左：3 自由度，右：9 自由度)

表 1　入力の自由度を変えた際の SD の変化

	3 自由度	9 自由度
SD	48.80	26.93

Q. 比較の際の関節角の範囲の決め方.

A. ロボットアームを制御する際に，実際に取り得る関節角の範囲に近いものとなっています．なぜならその方が，実際の制御に対して求めた近似式がより適当な式となるからです.

Q. 姿勢に線形性があるとはどういうことか.

A. 入力の一次式のみでアームの手先の姿勢を決める関節角を求めることができるということです．今回，近似式を用いようとしている制御式は，入力から出力であるロボットアームの各関節角速度(6 つ)を求め，それが手先の位置と姿勢を決める

ことになります．つまり，入力と出力の線形性が取れれば，その式から位置と姿勢の制御ができると考えられます．

Q. 近似する際の関節角の範囲によっては，アームを制御する範囲が変わってしまうのではないか．

A. 近似する関節角の範囲によってアームを制御できる範囲は変わります．なので，実装する際は，実際にアームが動く範囲内で近似する必要があります．

Q. 今後は視空間座標での線形近似を用いるのか．また，その理由．

A. この先，視空間と関節空間との線形性が示せれば，その線形近似式を用います．なぜなら，そうすることでキャリブレーションエラーに強い制御が実現できるからです．また，視空間座標はアームとカメラの位置関係を人間に近い状態にすれば，視空間と関節空間とに線形性が生じることが示されています．（ただし，3 自由度の位置制御に限ります)そこに，視空間座標を用いる理由があります．

Q. 線形近似が必要な理由．

A. ロボットパラメータに左右されない制御を実現するためです．そしてそれは，ロボットを使用する際に，パラメータの設定・計測の手間が省け，そこに研究する意義がありあます．通常の制御では入力から出力を求める際にロボットパラメータ(カメラ角やアーム長など)を用います．そうすると，その実際の値と推定値とに差が生じ，それが制御を不安定にする(これをキャリブレーションエラーと言います)要因になり得ます．これを防ぐ方法として，線形近似を用いることで，入力から出力を求める際の式にロボットパラメータを用いない式を使用でき，キャリブレーションエラーの回避に繋がります．

Q. 近似できる条件の求め方

A. ロボットパラメータを変化させ，その時々の近似精度を求め，最も良い結果のパラメータ条件を近似できる条件とするつもりです．

Q. 線形性は数式で判断できないのか．（テーラー展開を用いた方法）

A. 以下の方法により検証を行いました．

①あるロボットアームの関節角を指定します．

②実際の制御で用いる入力から出力を算出する式を求めます(式(2))．

③式(2)を$\boldsymbol{V} = \boldsymbol{a}$の周りでテーラー展開します(式(3))．

④式(2)と式(3)の第二項までの数式との差の二乗の平方根を計算します(式

(4)).

⑤指定された関節角の変域内で関節角を段階的に変化させ，その度に①～④を実行します.

⑥各関節角で得られた式(4)のdの平均を計算します.

以上の検証を視空間座標系と直交座標系で行い，それらの結果を表2と表3に示します.

ただし，以下に示す式は，簡略化のため入力が3次元の場合の式とします．実際の検証では入力を9次元としています(直交座標系は12次元です)．また，目標手先と初期手先との位置・姿勢誤差は，それぞれ，$(-0.2,-0.2,0.2)$ [m]とロールピッチヨー角で$(90,90,90)$[deg]とします.

また，この時の関節角の変域は，2パターン検証しました．1パターン目が，$20 \leq q_1 \leq 80$，$20 \leq q_2 \leq 80$，$20 \leq q_3 \leq 80$，$20 \leq q_4 \leq 80$，$20 \leq q_5 \leq 80$，$20 \leq q_6 \leq 80$であります．2パターン目が，$-100 \leq q_1 \leq -80$，$-100 \leq q_2 \leq -80$，$-10 \leq q_3 \leq 10$，$-100 \leq q_4 \leq -40$，$-10 \leq q_5 \leq 10$，$20 \leq q_6 \leq 40$であります．ただし，いずれも単位はdegreeであり，q_nはどこの関節かを表します(図2参照).

また，この検証で分かることとして，式(5)を参考に考えると，この値が0に近いほど式(2)がテーラー展開した式の第二項までとほぼ等価であることを表します．つまり，この値が0に近いほど，ロボットのパラメータを考慮した上で，元の制御式が線形な形で表せることを示すことになります.

今回の検証結果から，視空間座標を用いた場合の方が，画像座標(直交座標系)の時より，入力と出力の間に線形性があると言えます.

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_q^{-1}\boldsymbol{R}_{wc}\boldsymbol{J}_{Vimage}^{+}\dot{\boldsymbol{V}} \quad \cdots(2)$$

ただし，

$\boldsymbol{J}_q$:ロボットヤコビアン

$$\boldsymbol{J}_{Vimage} = \boldsymbol{J}_V^{-1}\begin{pmatrix} 1 & 0 & 0 & 0 & {}^cz & -{}^cy \\ 0 & 1 & 0 & -{}^cz & 0 & {}^cx \\ 0 & 0 & 1 & {}^cy & -{}^cx & 0 \end{pmatrix}$$

$\boldsymbol{J}_V$：視空間ヤコビアン

$({}^cx, {}^cy, {}^cz)$：カメラ座標系での手先特徴点座標

$\dot{\boldsymbol{V}}$:目標視空間座標と現在の視空間座標との差

$\boldsymbol{R}_{wc}$:回転変換行列

である.

$$f_{(\boldsymbol{V})} = \sum_{n=0}^{\infty} \frac{1}{n!} \left\{ \left(\frac{\partial}{\partial \boldsymbol{V}} \right)^n f_{(\boldsymbol{a})} \right\} (\boldsymbol{V} - \boldsymbol{a})^n \quad \cdots (3)$$

ただし，

$$\boldsymbol{V} = \begin{bmatrix} \gamma \\ \theta \\ \delta \end{bmatrix}$$

$$\boldsymbol{a} = \begin{bmatrix} \gamma_a \\ \theta_a \\ \delta_a \end{bmatrix}$$

である．

$$\boldsymbol{d} = \sqrt{(\dot{\boldsymbol{q}}' - \dot{\boldsymbol{q}})^T (\dot{\boldsymbol{q}}' - \dot{\boldsymbol{q}})} \quad \cdots (4)$$

表２　視空間座標系でのテーラー展開

	パターン１	パターン２
$\boldsymbol{a}$	(0,0,0)	(0,0,0)
$\boldsymbol{d}$	4.86×10^{-14}	2.17×10^{-15}

表３　直交座標系でのテーラー展開

	パターン１	パターン２
$\boldsymbol{a}$	(0,0,0)	(0,0,0)
$\boldsymbol{d}$	5.45×10^{-14}	2.33×10^{-15}

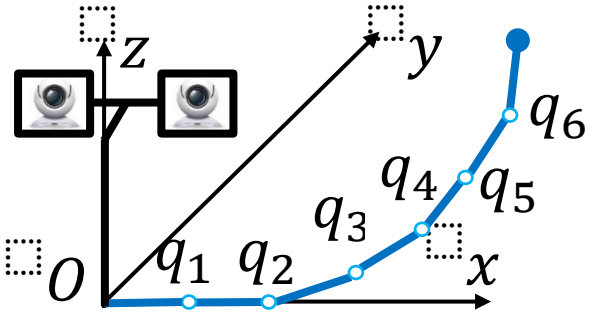

図２　シミュレーションモデル